REMARKS

Claims 1-21 are now pending. No claims stand allowed.

Claims 3, 9, 10, 11, 13, and 14 have been amended to further particularly point out and distinctly claim subject matter regarded as the invention. The text of claim 4 is unchanged, but its meaning is changed because it depends from amended claim 3.

New claims 15-21 have been added by this amendment and also particularly point out and distinctly claim subject matter regarded as the invention. Claims 15-21 are means-plus-function claims corresponding to method claims 1-7.

The specification and drawings have been amended to correct minor errors noted in the Office Action and otherwise. These corrections are of a clerical nature and do not add "new matter".

The First Objection to the Drawings

The Examiner states:

> The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they do not include the following reference sign(s) mentioned in the description: 26 and 28 mentioned on page 10, line 6.[1]

[1] Office Action dated November 20, 2002 at ¶ 1.

With this Amendment, FIG. 2A has been modified to include reference signs 26 and 28. Accordingly, withdrawal of the objection to the drawings is respectfully requested.

The Second Objection to the Drawings

The Examiner states:

> Furthermore, the drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference sign(s) not mentioned in the description: 105, 110, 115, 120, 125, and 130 in Figs. 4A & 4B, and 200, 205, 210, 215, and 220 in Figs. 6B & 6C.[2]

With this Amendment, FIGS. 4A and 4B have been modified to remove reference signs 105, 110, 115, 120, 125, and 130, and FIG. 6C has been modified to remove reference signs 210, 215, and 220. The specification has been amended to refer to reference signs 200 and 205 in FIG. 6B. Accordingly, withdrawal of the objection to the drawings is respectfully requested.

The Third Objection to the Drawings

The Examiner states:

> Next, the drawings are objected to as failing to comply with 37 CFR 1.84(p)(4) because reference character "12" has been used to designate both the Object class in Fig. 1B and 'a bus' with reference to Fig. 2A on page 10, line 4.[3]

With this Amendment, reference sign 12 of FIG. 1B has been changed to reference sign 5. Accordingly, withdrawal of the objection to the drawings is respectfully requested.

[2] Office Action at ¶ 2.
[3] Office Action at ¶ 3.

The Fourth Objection to the Drawings

The Examiner states:

> Finally, the drawings are objected to under 37 CFR 1.83(a). The drawings must show every feature of the invention specified in the claims. Therefore, the list structure of listing classes and interfaces separately in claim 3 must be shown or the feature(s) canceled from the claim(s). No new matter should be entered.[4]

With this Amendment, claim 3 has been amended to remove reference to an interface sublist. Accordingly, withdrawal of the objection to the drawings is respectfully requested.

Objections to the Claims

Claims 11 and 13 stand objected to under 37 CFR 1.75(c), as being of improper dependent form for failing to further limit the subject matter of a previous claim.[5] With this Amendment, claims 11 and 13 have been amended to place them in proper dependent form. Accordingly, withdrawal of the objection to claims 11 and 13 is respectfully requested.

The 35 U.S.C. § 112, Second Paragraph Rejection

Claim 9 stands rejected under 35 U.S.C. § 112, second paragraph, as being allegedly indefinite for failing to particularly point out and distinctly claim the subject matter applicant regards as the invention.[6] With this Amendment, claim 9 has been amended to depend from claim 8. Accordingly, withdrawal of the 35 U.S.C. § 112, second paragraph rejection is respectfully requested.

---

[4] Office Action at ¶ 4.
[5] Office Action at ¶ 5.
[6] Office Action at ¶ 6.

The First 35 U.S.C. § 103 Rejection

Claims 1, 2, 5, 6, 8, 9, 12, and 13 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Halstead et al.[7] [8] This rejection is respectfully traversed.

According to the M.P.E.P.,

> To establish a *prima facie* case of obviousness, three basic criteria must be met. First there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in the applicant's disclosure.[9]

Claim 1

Claim 1 recites:

> A method for representing an application programming interface (API) definition for an object-oriented library, said method comprising:
> creating a list of public elements in said library, each of said public elements including a sublist of all public related elements for the element; and
> storing said list.

The Examiner states:

> Referring to claim 1, Halstead discloses a system and method for representing an application programming interface definition for an object oriented library by generating a configuration file containing a hierarchical representation of the Java classes (tools) available for use in programming. See Figures 2-4 and the corresponding portions of the specification for this disclosure. Halstead's method is disclosed as follows:
> creating ['Step 411 creates' (Column 11, line 61)] a list ['configuration file 234' (Column 11, line 61. Also see Column 8, lines 38-44)] of public elements ['tools' (Column 5, line 48)], each of said public elements including a sublist of all public related elements for the element [lists the... parent child relationships' among tools (Column 8, lines 38-44. Also see column 2, line 65 et seq. and column 10, lines 8 36)]; and
> storing said list ['by copying' to the property store (Column 11, lines 61-63)].
> Halstead's 'tools' are Java classes, and more specifically are public classes (elements)

[7] U.S. Patent No. 6,230,318.
[8] Office Action at ¶ 7.
[9] Manual of Patent Examining Procedure (M.P.E.P) § 2143.

available for application programming. See Figure 2 and column 4, line 48 et seq. of Halstead's specification for this disclosure.[10]

The Applicant respectfully disagrees for the reasons set forth below.

I. Halstead et al. Does Not Teach All Claim Limitations

When evaluating a claim for determining obviousness, all limitations of the claim must be evaluated.[11]

Contrary to the Examiner's statement, Halstead et al. does not disclose creating a *list* of public elements in a library, where each of the public elements includes a *sublist* of all public related elements for the element. Rather, Halstead et al. discloses creating a configuration file.[12] Halstead et al. states:

> Configuration file 234 is a resource that specifies how tools are to be interconnected with each other. It is a text file that *lists the names of all tools in an application and the parent-child relationships among them.* The configuration file also specifies the layout of the user-interface elements such as tool interfaces and displayed visuals. Finally, this file contains persistent state data for any tool which needs it.[13]

Thus, the configuration file of Halstead et al. merely discloses listing parent-child relationships among tools in an application. Whereas claim 1 specifies that *each* of the public elements includes a sublist of *all* public related elements for the element. This per-public element list of *all* public related elements for an element is not disclosed in Halstead et al. Furthermore, nowhere does Halstead et al. indicate the tools are public. The Examiner is

[10] Office Action at ¶ 7.
[11] *In re Dillon*, 919 F.2d 688, 16 USPQ2d 1897 (Fed. Cir. 1990).
[12] Halstead et al. col. 8 lines 38-44.
[13] *Id.* (emphasis added)

reminded that the mere absence from a reference of an explicit requirement of a claim cannot be reasonably construed as an affirmative statement that the requirement is in the reference.[14] For this reason, the 35 U.S.C. § 103 rejection is unsupported by the art. Thus, no prima facie case of obviousness has been established and the 35 U.S.C. § 103 rejection should be withdrawn.

II. There Is No Basis in the Art for Combining or Modifying Halstead et al.

MPEP § 2143 provides:

> The mere fact that references *can* be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination.[15] (emphasis added)

Furthermore,

> Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching, suggestion or incentive supporting the combination.[16]

The Examiner contends:

> Halstead does not explicitly state that the configuration file (hierarchical list of tools) is created by listing public elements from a library as claimed. However, Halstead does state that programmers commonly use components (classes) taken from a library to build application programs. See column 1, lines 36-48 for this disclosure. This provides direct suggestion for building Halstead's configuration file by listing the public classes from an object oriented library.
>
> It would have been obvious to one of ordinary skill in the art at the time the invention was made to create Halstead's configuration file by listing the public classes from a Java library as the 'tools' available through the API to obtain the claimed method. One would have been motivated to do so because Halstead provides direct suggestion as described above.[17]

---

[14] *In re Evanega*, 829 F.2d 1110, 4 USPQ2d 1249 (Fed. Cir. 1987).
[15] *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990).
[16] *ACS Hospital Systems, Inc. v. Montefiore Hospital*, 732 F.2d 1572, 1577, 221 USPQ 929, 933 (Fed. Cir. 1984).
[17] Office Action at ¶ 7.

The Applicant submits there is no basis for modifying Halstead et al. The Examiner has stated the motivation to modify the reference is that Halstead et al. states programmers commonly use components taken from a library to build application programs.[18] However, Halstead et al. also states:

> The overall system also includes a number of individual components or tools collectively labeled 230, described in greater detail below. Each of these tools is a Java class. A number of operational Java classes 240 provide methods for running tools and other aspects of the system, although they are not themselves used as tools. It is important to note that the classes 230 and 240 may subclass each other in a normal manner so as to provide methods, data, and other facilities to each other. *This conventional static subclass hierarchy is entirely separate from the dynamic tool-tree hierarchy to be described below; the two hierarchies exist simultaneously, yet independently of each other.* Unless otherwise noted, the terms "hierarchy" and "tree" will refer to the dynamic tool-tree structure of the invention, rather than to the conventional static subclass structure of the classes 230 and 240.[19]

The Applicant respectfully submits that one of ordinary skill in the art would not have been motivated to create Halstead et al.'s configuration file by listing the public classes from a Java library as the tools available through the API because Halstead et al. teaches the conventional subclass hierarchy is entirely separate from the dynamic tool-tree hierarchy.

For this additional reason, the Examiner has failed to make a *prima facie* case of obviousness so the 35 U.S.C. § 103 rejection should be withdrawn.

III. Halstead et al. Is Not Properly Modifiable Because Such a Modification Would Destroy Its Intended Function.

[18] Office Action at ¶ 7.
[19] Halstead et al. at col. 5 lines 46-63. (emphasis added)

The CCPA had and the Federal Circuit have consistently held that when a § 103 rejection is based upon a modification of a reference that destroys the intent, purpose or function of the invention disclosed in the reference, such a proposed modification is not proper and the *prima facie* case of obviousness cannot be properly made.[20]

As mentioned above, the Halstead et al. reference requires the conventional static subclass hierarchy be entirely separate from the dynamic tool-tree hierarchy.[21] Halstead et al. also states that not all of the Java classes used in the invention are tools.[22] The modification proposed in the Examiner's rejection, i.e., creating Halstead et al.'s configuration file by listing the public classes from a Java library as the tools available through the API, would render Halstead et al. inoperable for its intended purposes. The so-modified Halstead. et al. construction would be inoperable for indicating the hierarchical relationship between tools because not all the Java classes are tools.

For this additional reason, the Examiner has failed to make a *prima facie* case of obviousness so the 35 U.S.C. § 103 rejection should be withdrawn.

IV. Halstead et al. Does Not Teach The Problem Or Its Source.

Halstead et al. is directed to the construction of entire application programs solely from small reusable components that can be substituted and reconfigured easily and quickly. More particularly, Halstead et al. discloses constructing an application program entirely from plug-together components or tools arrange in a tree hierarchy specified by a configuration file that can

[20] See e.g., *In re Gordon*, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984).
[21] Halstead et al. at col. 5 lines 56-58.

be *easily modified* to change the configuration *arbitrarily*.[23] Whereas the art of the present invention concerns representing an application programming interface (API) in an object-oriented system such that submerged hierarchies are enabled. Halstead et al. did not recognize the need for an API representation that sufficiently constrains particular implementations, while allowing them to define submerged hierarchies. Therefore, the Applicant submits there would be no motivation to modify the teachings of Halstead et al.

For this additional reason, the Examiner has failed to make a *prima facie* case of obviousness so the 35 U.S.C. § 103 rejection should be withdrawn.

Claim 2

Claim 2 depends from claim 1. The base claim being allowable, the dependent claim must also be allowable.

Claim 5

Claim 5 recites:

> A method for determining a program hierarchy, said method comprising:
> receiving an application programming interface (API) definition file for an object-oriented library, said API definition file including a list of public elements in said library, each of said public elements including a sublist of all public related elements for the element; and
> indicating a first public element is a direct parent of a second public element when said first public element is represented in the sublist for said second public element and said first public element is not represented in the sublist for any other public element listed in the sublist for said second public element.

The Examiner states:

---

[22] Halstead et al. at col. 5 lines 52-53.

> Referring to claim 5, Halstead discloses a method for determining a program hierarchy as claimed. Halstead's system copies (receives) a configuration file (application programming interface definition file) in step 411 shown in Figure 4. See Figure 4 and the corresponding portion of the specification for this disclosure. The format of the configuration file is exactly as claimed. See the discussion above regarding claim 1 for the details of this disclosure.[24]

The Applicant respectfully disagrees. Contrary to the Examiner's statement, the format of the configuration file is not exactly as claimed. Halstead et al. does not disclose receiving an application programming interface (API) definition file for an object-oriented library, where the API definition file includes a list of public elements in the library, and where *each* of the public elements includes a sublist of *all* public related elements for the element. As mentioned above with respect to claim 1, the configuration file of Halstead et al. merely discloses listing parent-child relationships among tools in an application.[25] The Examiner is referred to Halstead et al. at col. 10 lines 53-60. Nowhere does Halstead et al. disclose a *per-public element* sublist of *all* public related elements for the element.

The Examiner also states:

> Halstead does not explicitly disclose the second step of the claimed method. However, Halstead's system does analyze the hierarchy set forth in the configuration file when a resource is requested from a tool or when a new tool is added to the system. See column 11, lines 33-56 for this disclosure.
>
> Halstead does not explicitly disclose how a direct parent of a tool (public element) is found, but simply states that the system does find a tool's parent in order to traverse the hierarchy in the situations mentioned above.. However, looking at the structure of Halstead's configuration file described above with regard to claims 1 and 2, one can infer that the direct parent of a specific tool is represented in the sublist of that tool, but is not represented in the sublist of any other tools listed in that tool's sublist. In other words, in order to traverse Halstead's hierarchy, a first tool's direct parent can be found by searching that first tool's sublist to find the second tool that is not listed in the sublist for any other tool in the first tool's sublist.

---

[23] Halstead et al. at col. 2 lines 48-51. (emphasis added)
[24] Office Action at ¶ 7.
[25] Halstead et al. at col. 10 lines 58-60.

> It would have been obvious to one of ordinary skill in the art at the time the invention was made to program Halstead's system to traverse the configuration file's hierarchy by finding a first tool's direct parent by searching that first tool's sublist to find the second tool that is not listed in the sublist for any other tool in the first tool's sublist as claimed. One would have been motivated to do so because this method is easily inferred from the structure of the configuration file, and seems to be the only method for traversing the hierarchy possible.[26]

The Applicant respectfully disagrees. Again, the format the API definition file as claimed in claim 5 is entirely different than the configuration file disclosed in Halstead et al. Since the formats are different, it would not be obvious to search them the same way. It would also be nonobvious to search the configuration file using the method of claim 5 because parent-child relationship information may be obtained directly from the configuration file, without searching.

For this additional reason, the Examiner has failed to make a *prima facie* case of obviousness so the 35 U.S.C. § 103 rejection should be withdrawn.

Claim 6

Claim 6 depends from claim 5. The base claim being allowable, the dependent claim must also be allowable.

Claim 8

Claim 8 recites:

> A program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method to represent an application programming interface (API) definition for an object-oriented library, the method comprising:

[26] Office Action at ¶ 7.

creating a list of public elements in said library, each of said public elements including a sublist of all public related elements for the element; and
storing said list.

The Examiner states:

Referring to claim 8, the method of Halstead as discussed above with regard to claim 1 discloses the invention as claimed. Note that Halstead's method is implemented by computer executable instructions stored as program modules on a personal computer. See Figure 1 and the corresponding portion of the specification, specifically column 3, lines 30-48 for this disclosure.[27]

Claim 8 includes limitations similar to claim 1. Claim 1 being allowable, claim 8 must also be allowable.

Claim 9

Claim 9 as amended recites:

The program storage device of claim 8 wherein
said library is a Java™ package;
said public elements comprise classes and interfaces; and
said public related elements comprise public superclasses and public superinterfaces of said classes and said interfaces.

The Examiner states:

Referring to claim 9, the system and method of Halstead as discussed above with regard to claim 2 discloses the invention as claimed. See also the discussion of claim 8 for the details of this disclosure.[28]

Claim 9 includes limitations similar to claim 2. Claim 2 being allowable, claim 9 must also be allowable.

[27] Office Action at ¶ 7.
[28] Office Action at ¶ 7.

Claim 12

Claim 12 recites:

A program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method to determine a program hierarchy, the method comprising:
receiving an application programming interface (API) definition file for an object-oriented library, said API definition file including a list of public elements in said library, each of said public elements including a sublist of all public related elements for the element; and
indicating a first public element is a direct parent of a second public element when said first public element is represented in the sublist for said second public element and said first public element is not represented in the sublist for any other public element listed in the sublist for said second public element.

The Examiner states:

Referring to claim 12, the system and method of Halstead as discussed above with regard to claim 5 discloses the invention as claimed. See also the discussion regarding claim 8 for the details of this disclosure.[29]

Claim 12 includes limitations similar to claim 5. Claim 5 being allowable, claim 12 must also be allowable.

Claim 13

Claim 13 as amended recites:

The program storage device of claim 12 wherein
said library is a Java™ package;
said public elements comprise classes and interfaces; and
said public related elements comprise public superclasses and public superinterfaces of said classes and said interfaces.

The Examiner states:

[29] Office Action at ¶ 7.

> Referring to claim 13, the system and method of Halstead as discussed above with regard to claim 6 discloses the invention as claimed. See also the discussion regarding claim 8 for the details of this disclosure.[30]

Claim 13 includes limitations similar to claim 6. Claim 6 being allowable, claim 13 must also be allowable.

For these reasons, the Examiner has failed to make a *prima facie* case of obviousness so the 35 U.S.C. § 103 rejection as to claims 1, 2, 5, 6, 8, 9, 12, and 13 should be withdrawn.

In view of the foregoing, it is respectfully asserted that the claims are now in condition for allowance. It is respectfully requested that the 35 U.S.C. § 103 rejection of claims based on Halstead et al. be withdrawn.

The Second 35 U.S.C. § 103 Rejection

Claims 3, 4, 10 and 11 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Halstead et al. in view of the Specification.[31] This rejection is respectfully traversed.

Claim 3

Claim 3 as amended recites:

> A method for representing an application programming interface (API) definition for an object-oriented library, said method comprising:
> creating a public list including all public classes and interfaces in defined in said library, said public list including a class sublist for each of said public classes, each said class sublist including all direct and indirect superclasses of a class; and

[30] Office Action at ¶ 7.
[31] Specification at p. 2 line 18 to p. 3 line 5.

storing said list.

The Examiner states:

> Referring to claim 3, the method of Halstead as discussed above with regard to claims 1 and 2 discloses the method as claimed. See the discussion above regarding claims 1 and 2 for the details of this disclosure. Halstead's configuration file does not list the classes separately from the interfaces as claimed. Instead, the configuration file only lists the tools and their parent tools in a sublist for each tool. Each tool then has its class and interfaces defined within the tool definition. This provides direct suggestion for separating the classes from the interfaces.
>
> Applicant's admitted prior art of page 2, line 18 - page 3, line 5 discloses that API definition files typically include the available classes and interfaces separately, along with the immediate superclass and superinterface for each class and interface.
>
> Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Halstead's system such that when building the configuration file, the classes and interfaces for each tool would be listed separately instead of just the name of the tool, and the superclasses and superinterfaces would be listed in the sublists instead of just the parent tool names. One would have been motivated to do so because this is the typical format of such a file, as disclosed by applicant's admitted prior art.[32]

The Applicants respectfully disagree. The arguments made with respect to claim 1 apply here as well. Contrary to the Examiner's statement, Halstead et al. in combination with Applicant's admitted prior art does not disclose creating a public list including all public classes and interfaces defined in the library, where the public list includes a class sublist for each of the public classes and where *each* class sublist includes *all* direct *and indirect* subclasses of a class. As the Examiner indicated, the Applicant's admitted prior art of page 2, line 18 – page 3 line 5 discloses that API definition files typically include the available classes and interfaces separately, along with the *immediate* superclass and superinterface for each class and interface.[33] However, claim 3 specifies each class sublist includes all direct *and indirect* subclasses of a

[32] Office Action at ¶ 8.
[33] Office Action at ¶ 8.

class. This is not disclosed or suggested in the Applicant's prior art. Nor is it disclosed or suggested in Halstead et al.

For this additional reason, the 35 U.S.C. § 103 rejection is unsupported by the art. Thus, no prima facie case of obviousness has been established and the 35 U.S.C. § 103 rejection should be withdrawn.

Claim 4

Claim 4 depends from claim 3. The base claim being allowable, the dependent claim must also be allowable.

Claim 10

Claim 10 as amended recites:

> A program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method to represent an application programming interface (API) definition for an object-oriented library, the method comprising:
> creating a public list including all public classes and interfaces in defined in said library, said public list including a class sublist for each of said public classes, each said class sublist including all direct and indirect superclasses of a class; and
> storing said list.

The Examiner states:

> Referring to claim 10, the system and method of Halstead in view of applicant's admitted prior art as discussed above with regard to claim 3 discloses the invention as claimed. See also the discussion regarding claim 8 for the details of this disclosure.[34]

[34] Office Action at ¶ 8.

Claim 10 includes limitations similar to claim 3. Claim 3 being allowable, claim 10 must also be allowable.

Claim 11

Claim 11 as amended recites:

The program storage device of claim 10 wherein said library is a Java™ package.

The Examiner states:

> Referring to claim 11, the system and method of Halstead in view of applicant's admitted prior art as discussed above with regard to claim 4 discloses the invention as claimed. See also the discussion regarding claim 8 for the details of this disclosure.[35]

Claim 11 includes limitations similar to claim 4. Claim 4 being allowable, claim 11 must also be allowable.

For these reasons, the Examiner has failed to make a *prima facie* case of obviousness so the 35 U.S.C. § 103 rejection as to claims 3, 4, 10, and 11 should be withdrawn.

In view of the foregoing, it is respectfully asserted that the claims are now in condition for allowance. It is respectfully requested that the 35 U.S.C. § 103 rejection of claims based on Halstead et al. in view of the Specification be withdrawn.

The Third 35 U.S.C. § 103 Rejection

[35] Office Action at ¶ 8.

Claims 7 and 14 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Halstead et al. in view of Gossain et al.[36] [37] This rejection is respectfully traversed.

Claim 7

Claim 7 recites:

The method of claim 5, further comprising:
comparing a first program hierarchy reconstructed from a first API definition file with a second program hierarchy reconstructed from a second API definition file; and
indicating an error when said first program hierarchy is inconsistent with said second program hierarchy.

The Examiner states:

> Referring to claim 7, Halstead's method of analyzing the hierarchy of a configuration file as discussed above with regard to claim 5 does not disclose the steps of comparing two hierarchies and indicating an error when they are inconsistent as claimed. However, Halstead does disclose a need to maintain consistency of the hierarchy when the API is changed or when a new tool is added. See column 2, lines 15-30 and column 10, lines 45-52 for this disclosure. This provides suggestion for examining the hierarchy of an API with an expected hierarchy to maintain consistency.
>
> Gossain discloses a method for testing the inheritance hierarchy of an object oriented class structure by comparing the active hierarchy to a test hierarchy stored within the system. See the Figure and the Detailed Description of the Drawing section for this disclosure. Refer specifically to column 3, lines 6-14. Gossain teaches the two claimed steps as follows: Comparing [step 18] a first program hierarchy [hierarchy of class under test (11)] with a second program hierarchy [test class hierarchy (12)]; and Indicating an error [Column 3, lines 9-10] when said first program hierarchy is inconsistent ['when a difference between the current state and expected state ... is detected' (Column 3, lines 7-8)] with said second program hierarchy.
>
> It would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate Gossain's method for testing class hierarchies into Halstead's system such that the system would compare the hierarchy reconstructed from a configuration file with the hierarchy reconstructed from a test configuration file, and indicate an error when the two hierarchies were inconsistent. One would have been motivated to do so because of Halstead's suggestion described above.[38]

[36] USP 5,974,255.
[37] Office Action at ¶ 9.
[38] Office Action at ¶ 9.

The Applicants respectfully disagree. Claim 7 depends from claim 5. The base claim being allowable, the dependent claim must also be allowable. Furthermore, contrary to the Examiner's statement, neither Halstead et al. nor Gossain et al. discloses comparing a first program hierarchy *reconstructed from a first API definition file* with a second program hierarchy *reconstructed from a second API definition file*. There is nothing to reconstruct in both Halstead et al. and Gossain et al., since hierarchical information is already present. Whereas the hierarchical information is not immediately discernable in the method of independent claim 5 and dependent claim 7, thus requiring the program hierarchies to be reconstructed.

For this additional reason, the 35 U.S.C. § 103 rejection is unsupported by the art. Thus, no prima facie case of obviousness has been established and the 35 U.S.C. § 103 rejection should be withdrawn.

Claim 14

Claim 14 as amended recites:

> The program storage device of claim 12 wherein said method further comprises:
> comparing a first program hierarchy reconstructed from a first API definition file with a second program hierarchy reconstructed from a second API definition file; and
> indicating an error when said first program hierarchy is inconsistent with said second program hierarchy.

The Examiner states:

> Referring to claim 14, the system and method of Halstead in view of Gossain as described above with regard to claim 7 discloses the invention as claimed. See also the discussion above regarding claim 8 for the details of this disclosure.[39]

[39] Office Action at ¶ 9.

Claim 14 includes limitations similar to claim 7. Claim 7 being allowable, claim 14 must also be allowable.

For these reasons, the Examiner has failed to make a *prima facie* case of obviousness so the 35 U.S.C. § 103 rejection as to claims 7 and 14 should be withdrawn.

In view of the foregoing, it is respectfully asserted that the claims are now in condition for allowance. It is respectfully requested that the 35 U.S.C. § 103 rejection of claims based on Halstead et al. in view of Gossain et al. be withdrawn.

The attached page is captioned "**Version with markings to show changes made.**"

Request for Allowance

It is believed that this Response places the above-identified patent application into condition for allowance. Early favorable consideration of this application is earnestly solicited.

If, in the opinion of the Examiner, an interview would expedite the prosecution of this application, the Examiner is invited to call the undersigned attorney at the number indicated below.

Respectfully submitted,
THELEN REID & PRIEST, LLP

Dated: February 26, 2003

John P. Schaub
Reg. No. 42,125

THELEN REID & PRIEST LLP
P. O. Box 640640
San Jose, CA 95164-0640
Tel: (408) 292-5800
#120998V1

Version With Markings To Show Changes Made

3. (Amended Once) A method for representing an application programming interface (API) definition for an object-oriented library, said method comprising:

creating a public list including all public classes and interfaces in defined in said library, said public list including a class sublist for each of said public classes, each said class sublist including all direct and indirect superclasses of a class[, said public list including an interface sublist for each of said public interfaces, each said interface sublist including all direct and indirect public superinterfaces of an interface]; and

storing said list.

9. (Amended Once) The program storage device of claim [7] 8 wherein

said library is a Java™ package;

said public elements comprise classes and interfaces; and

said public related elements comprise public superclasses and public superinterfaces of said classes and said interfaces.

10. (Amended Once) A program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method to represent an application programming interface (API) definition for an object-oriented library, the method comprising:

creating a public list including all public classes and interfaces in defined in said library, said public list including a class sublist for each of said public classes, each said class sublist including all direct and indirect superclasses of a class[, said public list including

an interface sublist for each of said public interfaces, each said interface sublist including all direct and indirect public superinterfaces of an interface]; and

storing said list.

11. (Amended Once) The program storage device of claim [9] 10 wherein said library is a Java™ package.

13. (Amended Once) The program storage device of claim [11] 12 wherein

said library is a Java™ package;

said public elements comprise classes and interfaces; and

said public related elements comprise public superclasses and public superinterfaces of said classes and said interfaces.

14. (Amended Once) The program storage device of claim 12[, further comprising] wherein said method further comprises:

comparing a first program hierarchy reconstructed from a first API definition file with a second program hierarchy reconstructed from a second API definition file; and

indicating an error when said first program hierarchy is inconsistent with said second program hierarchy.

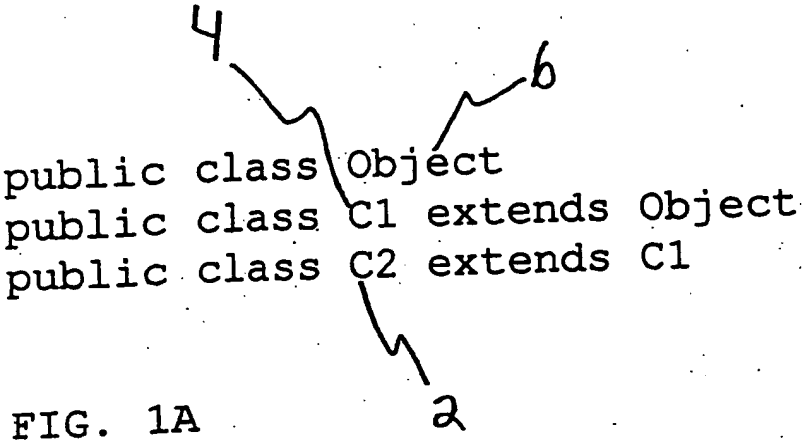

FIG. 1A

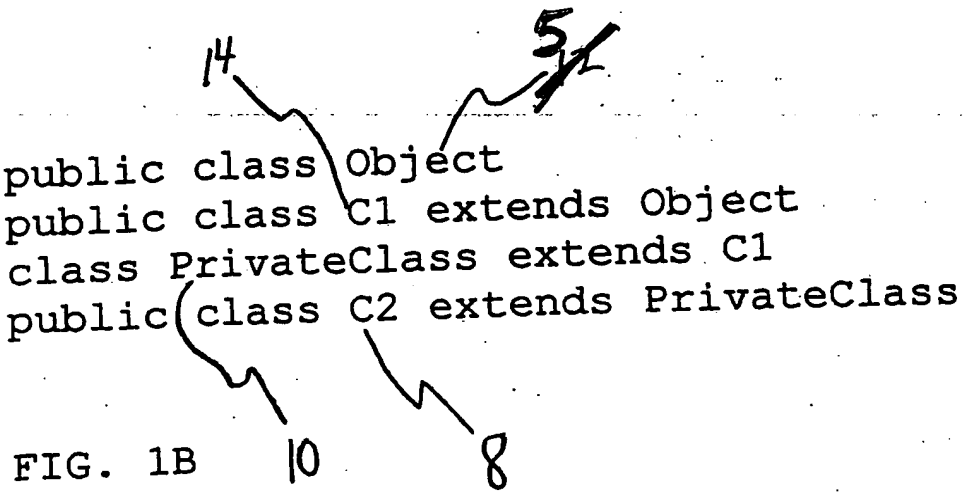

FIG. 1B

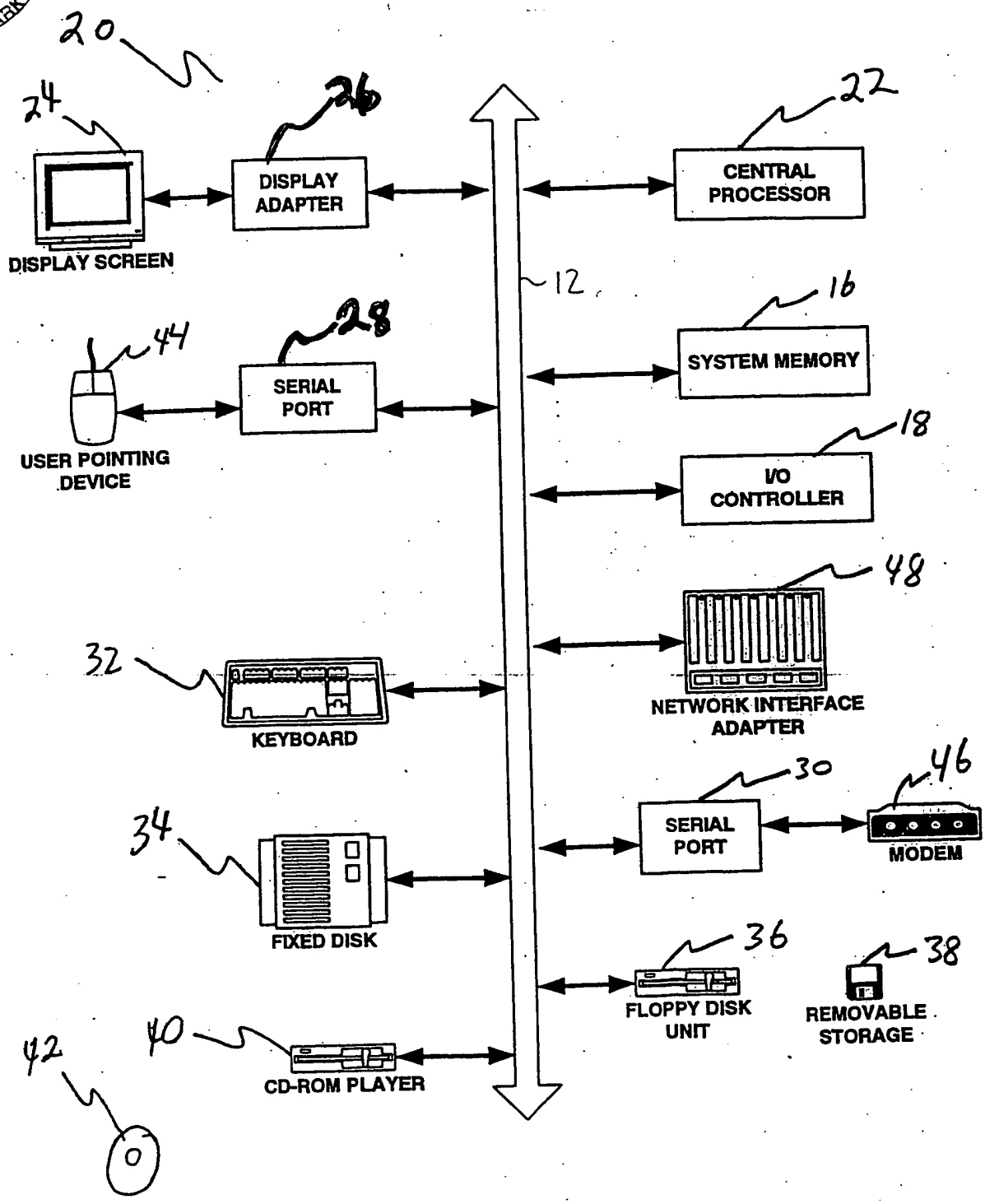

Fig. 2A

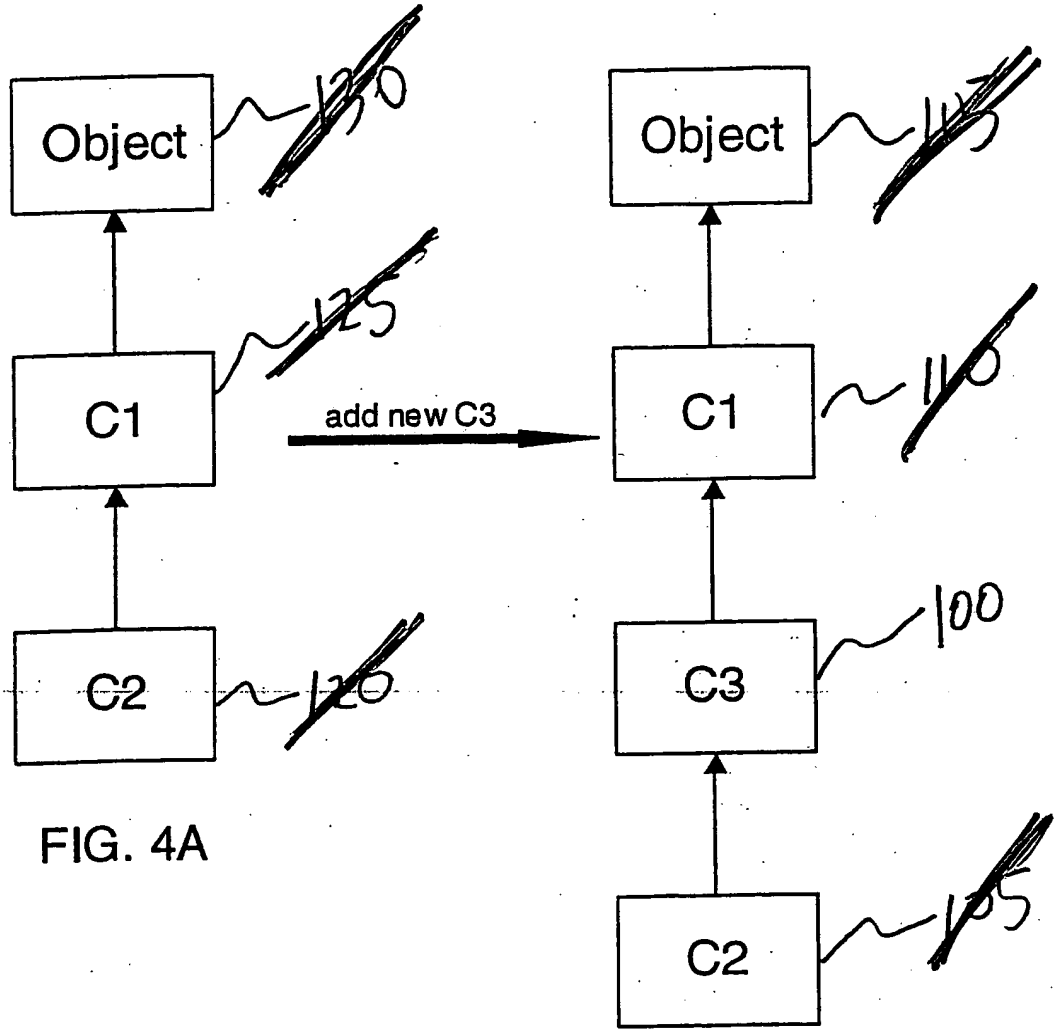

FIG. 4A

FIG. 4B

```
public class C3 extends Object
public class C4 extends C2
```

200

205

FIG. 6B

```
public class C3
    superclasses = Object
public class C4
    superclasses = C2, C1, Object
```

FIG. 6C